[MySym] - 自定义 Sym 论坛编辑器中 emoji 列表提示

作者：ZephyrJung
原文链接：https://ld246.com/article/1470285140911
来源网站：链滴

通过问D大，得知ctrl+/的代码在common.js里，搜索emoji关键字找到emojiString，此处存放了所emoji名称的字符串（老长了D大怎么做到的……）

于是在此处添加ajax方法，以从后台获取该用户相关的emoji列表：

```
$.ajax({
  async: false,
  url: Label.servePath + "/users/emotions",
  type: "GET",
  success: function (result) {
      console.log('ajax:'+result.emotions);
      emojString=result.emotions;
  }
});
```

其实我不太清楚async的含义，也懒得管，能用就行0.0~

这里其实用了一个CodeMirror的插件（似乎是做代码自动补全的，D大将它用到了表情符号上 1）

鉴于这块代码内也有其他ajax并且调用了UserProcessor（类似于Controller），我也将emotion获的controller写在了这里（因为是获取该用户的emoji，所以也不算打乱，应该是吧）

```
@RequestProcessing(value = "/users/emotions", method = HTTPRequestMethod.GET)
public void getEmotions(final HTTPRequestContext context, final HttpServletRequestrequest,f
nal HttpServletResponse response) throws Exception {
   String emotions;
   context.renderJSON();//context包含request和resposne对象，不知道是否一致
   final JSONObject currentUser = (JSONObject) request.getAttribute(User.USER); //获到的包
用户信息的JSON对象
   final String userId = currentUser.optString(Keys.OBJECT_ID); //取得用户ID1470108766562
   emotions=emotionQueryService.getEmotions(userId);//保存
   context.renderJSONValue("emotions",emotions);
}
```

这段代码是拷贝的其他方法里的再做修改，注释是我通过调试读取到的值，以明白这段代码是用来做么的（这段时间工作的新技能，不必探查源码考究代码的功能，通过调试知道其能获取什么值就直接来用好了）

可以看到，这里建立了emotionQueryService，需要如下引入：

```
@Inject
private EmotionQueryService emotionQueryService;
```

虽然这个ajax是void类型的，不过可以通过context.renderJSONValue方法将值带回去（这个是向D教类似功能的代码得出的结论）

于是，进入getEmotions方法。

在论坛上问过D大Service，Repository，Cache，以及Model类的关系（查看帖子）

如D所言，Repository就是执行数据库查询操作的DAO，不过我得说，Service也有Query代码，这于分层不够明确:P

Service是服务类，按我的理解就是调用Repository以完成Processing需要的动作，Cache，D大的解是放入内存，有些操作就是先对内存数据进行操作，完后再写回数据库的，如此大概会提高速度（积就是如此，以至于直接通过该数据库库字段的话，积分不会变化，需要重启服务才能生效）

于是，我照着原有规则，写了一个EmotionQueryService和EmotionMgmtService，目前只用到了

者：

完整代码由于包含完整emoji列表比较长，就不贴了，关键代码如下：

```
try {
  String emojis=emotionRepository.getUserEmotions(userId);
  if(emojis!=null&&emojis.length()!=0)
        return emojis;
} catch (final RepositoryException e) {
LOGGER.log(Level.ERROR, e.getMessage());
  return allEmojis;
}
```

这块没什么可说的了，关键在于Repository里面：

```
@Repository
public class EmotionRepository extends AbstractRepository {
    public EmotionRepository() {
        super("emotions");//此处据我推测应该与repository.json里的name有关
    }
    public String getUserEmotions(final String userId) throws RepositoryException {
        final PropertyFilter pf=new PropertyFilter(Emotion.EmotionUser, FilterOperator.EQUAL, u
erId);//定义查询条件
        final Query query = new Query().setFilter(pf);//添加查询条件
        final JSONObject result = get(query);//执行查询
        final JSONArray array = result.optJSONArray(Keys.RESULTS);
        if (0 == array.length()) {
            return null;
        }
        String resultString="";
        try {

            for(int i=0;i<array.length();i++){
                resultString+=array.optJSONObject(i).get("emotionName").toString();
                if(i!=array.length()-1)
                    resultString+=",";
            }
        } catch (JSONException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return resultString;
    }
}
```

Query或者Filter的具体逻辑无需关心，直接可用（一般情况下应该很容易吧，再复杂我还没有考虑过我这里只需要通过userid获取到所有结果就行）

这个项目json处处可见，此处亦然，只定义Repository还不行，还需要在repository.json里添加数据的对应：

```
{
    "name": "emotions",
    "keys": [
        {
            "name": "oId",
```

```
            "type": "String",
            "length": 19
        },
        {
            "name": "user_old",
            "type": "String",
            "length": 19
        },
        {
            "name": "emotionName",
            "type": "String",
            "length": 25
        },
        {
            "name": "emotionSort",
            "type": "int"
        },
        {
            "name": "emotionType",
            "type": "int"
        }
    ]
}
```

第一个name疑似repository的super里传入的参数，其他的则对应数据表的列名了。

如此，emoji快捷键调出来的emoji列表就掌握在自己手中了，还有相应的管理等功能，尚未完成，装要趁早，所以先写了个这个流水账。实际开发的流程其实是逆推过来的，先模拟了service，repositor等再反过来向上写。